NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN APPLICATION OF ROLE-BASED ACCESS CONTROL IN AN ORGANIZATIONAL SOFTWARE PROCESS KNOWLEDGE BASE

by

William A. Windhurst

June 2001

Thesis Advisor: James Bret Michael
Associate Advisor: John Osmundson

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 2001	Master's Thesis

4. TITLE AND SUBTITLE
An Application Of Role-Based Access Control In An Organizational Software Process Knowledge Base

5. FUNDING NUMBERS

6. AUTHOR(S)
Windhurst, William A.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

The Organizational Software Process Knowledge Base (OSPKB) is the repository of an organization's software process, product performance, quality metrics, and lessons learned. The knowledge is maintained on a project-by-project basis, as well as by business domain. The OSPKB contains sensitive data and information that needs to be protected from unauthorized disclosure or modification. In this thesis, we address the challenge of controlling access to the data and information stored in the OSPKB. In particular, we investigate approaches to applying role-based access control (RBAC) to OSPKB applications.

14. SUBJECT TERMS
Project Management, Software Process Management, Role-Based Access Control, Security

15. NUMBER OF PAGES
68

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited

AN APPLICATION OF ROLE-BASED ACCESS CONTROL IN AN ORGANIZATIONAL SOFTWARE PROCESS KNOWLEDGE BASE

William A. Windhurst
B.S., Coleman College, 1982

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2001

Author: William A. Windhurst
William A. Windhurst

Approved by: James Bret Michael
James Bret Michael, Thesis Advisor

John Osmundson
John Osmundson, Associate Advisor

Luqi, Chairman
Software Engineering Curriculum

Dan C Boger
Dan Boger, Chairman
Department of Computer Science

ABSTRACT

The Organizational Software Process Knowledge Base (OSPKB) is the repository of an organization's software process, product performance, quality metrics, and lessons learned. The knowledge is maintained on a project-by-project basis, as well as by business domain. The OSPKB contains sensitive data and information that needs to be protected from unauthorized disclosure or modification. In this thesis, we address the challenge of controlling access to the data and information stored in the OSPKB. In particular, we investigate approaches to applying role-based access control (RBAC) to OSPKB applications.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

I would like to express my sincere appreciation to Dr. James Bret Michael and Dr. John Osmundson for their unfailing support and encouragement, but most of all for their patience with me. I would also like to thank my family and friends for putting up with me while I conducted the research and wrote and rewrote this thesis. Without their support this thesis would never have been completed.

I. SOFTWARE PROCESS MANAGEMENT

A. PROCESS MAINTENANCE

The software process is the set of tools, methods, and practices used to produce software products. The objectives of software process management are to produce products according to plan while simultaneously improving the organization's capability to produce better products [1].

The objectives of process management are to ensure that the processes within the organization are performing as expected, to ensure that defined processes are being followed, and to make improvements to the processes to meet business objectives.

Process management principles are based on statistical process control, which has been used with great success in many fields. A process is said to be stable or under statistical control when future performance of the process is predictable within established statistical limits [2]. Once a process is under control, sustaining activities must be undertaken to forestall the effects of entropy. Without these activities, processes can fall victim to the forces of uncontrolled change or disuse, or deteriorate into unmanageable states [3].

There are four responsibilities that are central to process management:

- Defining the process
- Measuring the process (collect and store process-performance data)
- Controlling the process (ensure the variability is stable so that results are predictable)
- Improving the process

These four areas of responsibility are shown as boxes in Figure 1-1. Process execution is shown as a circle because execution is not a process management responsibility.

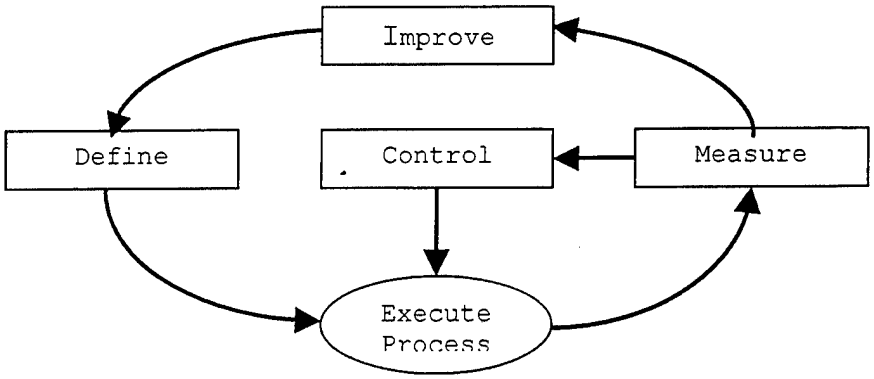


Figure 1-1 Four Responsibilities of Process Management From Ref. [3]

B. PROCESS MONITORING AND CONTROL

The Organizational Process Knowledge Base (OSPKB) is concerned with collection, storage of project and process performance data from all business-area domains. Data is collected during the enactment of the software engineering processes. Data definitions, context descriptions, and direct measurement data are recorded. Measured values are linked to their respective measurement definitions, rules, practices, and project management tools used. Entities and attributes are linked to measured values. Measured values are tied to the environment context in which they were collected (e.g., product, environment, process descriptors; process and project status; time and place measured; measurement methods).

Process control charts are constructed from the measurements. Control charts have been used in industry since the 1920s. As Montgomery points out, there are at least five reasons for their popularity [4]:

1. Control charts are a proven technique for improving productivity.
2. Control charts are effective in defect prevention.

3. Control charts help prevent unnecessary process adjustments.
4. Control charts provide diagnostic information.
5. Control charts provide information about process capability.

C. PROCESS ENACTMENT

Software engineering processes are enacted by humans or intelligent software agents acting as proxies for human agents. Automated processes can be instrumented for automated collection of measurement data. Human-enacted processes require measurement data to be manually collected at the appropriate points during process operation by the human agents enacting the process.

D. PROBLEM STATEMENT

Enactment of a software management process results in the generation of data and information about people and products. Data collected about people (e.g., project managers and software developers), in its raw form or even in a statistical form, may be traceable back to an individual. Some of that data and information may require protection from unauthorized disclosure, both to parties internal and external to the organization. For example, information about an employee's performance of activities may be protected by the organization's privacy policy. The risk of releasing such information to parties without a bonafide need to know may also be intolerable for the organization, especially if such release could result in litigation, unnecessary degradation of employee morale, or loss of corporate or customer goodwill.

Uncontrolled disclosure of data and information about the development and maintenance of a software product could result in the compromise of proprietary data and

information, possibly to the organization's competitors. In addition, a software process can involve the participation of multiple organizations. These organizations may expect, through contractual (i.e., legally binding) agreements, that all parties to the process protect all or specific portions of the contents of the OSPKB. Failure by one of the parties to do so could result in a loss of willingness by one or more of the parties to share information about their process, damage to business-to-business relationships, or economic damages (e.g., due to litigation or loss of trade secrets). Thus it is necessary to have an access control mechanism to enforce the disclosure policies of the parties participating in the enactment of a software process.

In order to determine the requirements for such a mechanism, one must first specify the access control policy. Artifacts generated by the enactment of the software process are traceable to the role played by a team or individual that produced the artifact. This leads us to believe that the policy-specification model that we produce should lend itself to representing access to the contents of the OSPKB based on both the role of the requester (i.e., human or agent initiating a query or update operation) and the roles associated with the data to be accessed via the query or update operation.

E. HYPOTHESIS

Our hypothesis is that a role-based access control model can be used to specify the types of disclosure policies associated with an OSPKB.

F. SCOPE

We test our hypothesis using the actual and anticipated flows of information to and from the OSPKB. The US Navy's Space and Naval Warfare Systems Center San Diego (SSC San Diego) is in the process of developing the OSPKB. We limit our analysis to a case study of a subset of the roles and information flows associated with the software process. We also confine our investigation to a subset of the documented access control policies of SSC San Diego and other parties with which SSC San Diego cooperates to produce software.

G. APPROACH

We analyze the OSPKB with respect to the organizational roles supplying data to or drawing data from the OSPKB. The results of the analysis forms the basis for engineering a hierarchy of organizational roles associated with the OSPKB. Next, we represent the roles, along with the access control policy, for one of the OSPKB applications, using the semantics of a role-based access control model. We assess the extent to which the resulting model corresponds to the roles and relationships that formed the basis of our case study. We conclude our analysis by assessing the advantages and limitations of applying role-based access control in the context of the OSPKB.

THIS PAGE INTENTIONALLY LEFT BLANK

II. ROLE BASED ACCESS CONTROL AND UNIFIED MODELING LANGUAGE

A. REVIEW OF RBAC MODEL

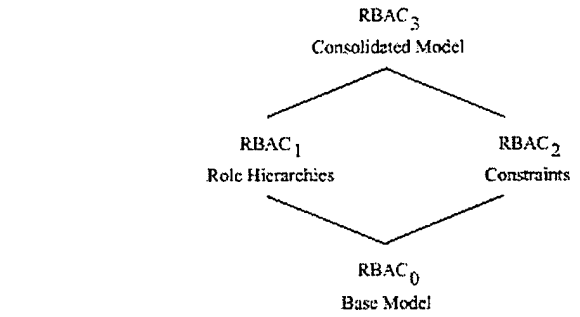
Role-based access control (RBAC) is founded on linking access permissions to each distinct role [5]. A permission is an authorization to access one or more objects in the system. The nature of a permission depends on the implementation details of the system. Each system protects objects of the abstraction it implements. For example, an operating system protects such entities as files, directories, devices and ports, according to access mode, such as read, write and execute. A relational database management system such as an OSPKB protects relations, records, attributes, and views, according to access mode, such as SELECT, UPDATE, DELETE, and INSERT.

A user in the context of the RBAC model can be a human being or an intelligent software agent acting as a proxy for the human [5].

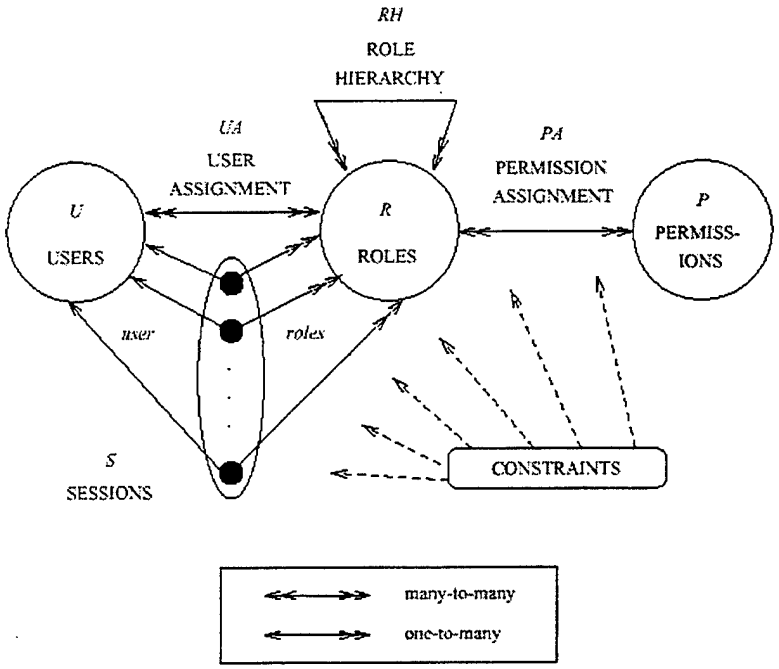
A role is a job function or job title within the organization with associated details concerning the authorities and responsibilities assigned to a member of the role. The particular set of users and permissions associated with a role can be short lived. The role is generally more stable because organizational activities and functions are relatively stable [5]. In a software engineering organization, some typical roles are programmer, designer, tester, project manager, and project planner. Each of these roles may differ from one another in terms of the permissions needed by the actor playing that role to carry out its responsibilities with respect to the OSPKB.

RBAC96 defines a family of four models to fully describe RBAC. $RBAC_0$ is the basic RBAC model encompassing

three constructs: users, roles, and permissions. $RBAC_1$ adds the concept of role hierarchies, that is situations where roles can inherit permissions from other roles. $RBAC_2$ Adds the concept of constraints; the constraints impose restrictions on acceptable configurations of users, roles, and permissions. The consolidated model, $RBAC_3$, includes $RBAC_1$ and $RBAC_2$ and, by transitivity $RBAC_0$ [5]. The relationship between the models is shown in Figure 2-1(a) and the consolidated model $RBAC_3$ is portrayed in Figure 2-1(b).



(a) Relationship among RBAC96 models



(b) The $RBAC_3$ model

Figure 2-1 The RBAC 96 Family of Models from Ref. [5]

B. RBAC ARCHITECTURE

RBAC architecture has been represented as a three-tier structure based on the classic ANSI/SPARC relational database architecture [5]. This architecture describes external or user views, a conceptual or community view, and internal or implementation dependent views. The architecture is representative as RBAC is concerned with the meaning and control of access data (i.e., data used to control access to the actual data of the organization)[5]. The three-tier architecture is shown in Fig 2-2

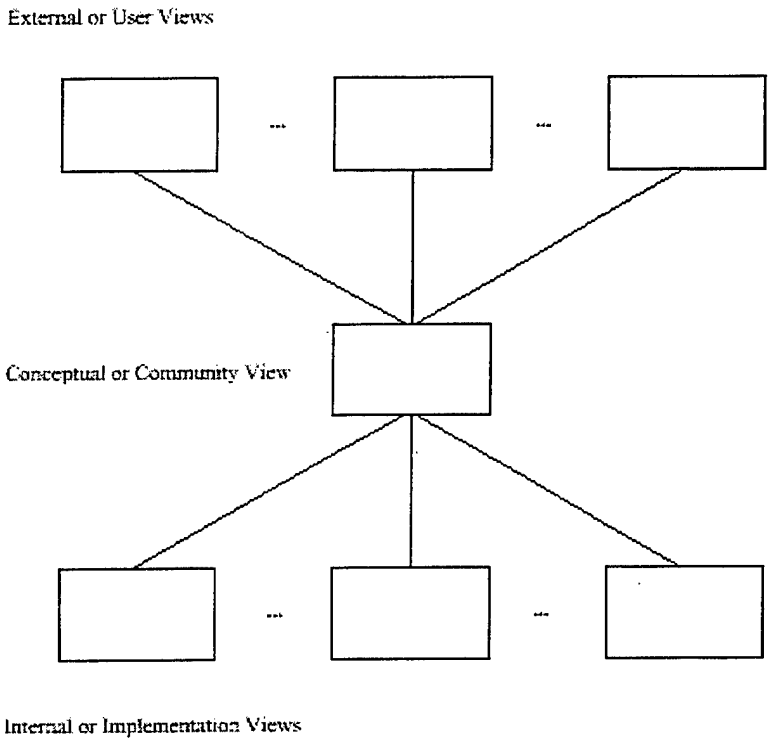


Figure 2-2 A three Tier Architecture for RBAC from Ref.[5]

Each of the external views aggregates one perspective of the community view relevant to a particular context. Each of the internal views likewise aggregates the implementation details of the community view that is implemented on a particular platform.

C. UML MODELING OF RBAC

The role-engineering functions of RBAC can be modeled in UML. Epstein and Sandhu describe the modeling approach in their paper "Towards A UML Based Approach to Role Engineering" [6]. UML was used to model Role Engineering in the context of the Role-Based Access Control Framework for Network Enterprises (FNE), developed by Thomsen, O'Brien, and Bogle [7]. FNE is based on the divide-and-conquer principle: No one person in responsible for the security management of an entire system. Different groups administer the seven abstract layers of the FNE. The first four layers form the foundation of the model, and are administered by the application developer. The upper three layers are the enterprise layer and are administered by the local system administrator. A description of the layers and associated UML model elements follows:

1. Objects: The main component of the first layer is an object. An object has a name and a set of public methods that can be used to access the object. The methods constitute the permissions necessary to perform actions on the object. The object attributes can be used as conditions for constraints. See Figure 2-3.

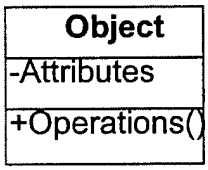


Figure 2-3 Object and Methods (Layer 1) after Ref. [6]

2. Object Handles: This layer is a collection of named objects and associated access methods called a handle set. See Figure 2-4.

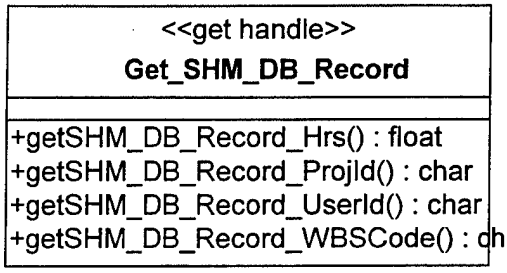


Figure 2-4 Object Handle (Layer 2) after Ref. [6]

3. Application Constraints: Application constraints must be satisfied to gain access to the methods in a handle set. See Figure 2-5

Operation name:
Get Project Manager
Public member of: Project Record
Preconditions:
The project is the project of the project manager.

Figure 2-5 Example of an Application Constraint (Layer 3) after Ref. [6]

4. Application Keys: An application key associates a role with objects, data records, and methods. Application keys are either specific to a role (e.g., Project X manager), or are abstract (e.g., generic manager). Any abstract key is considered to be mapped to an abstract role. See Figure 2-6.

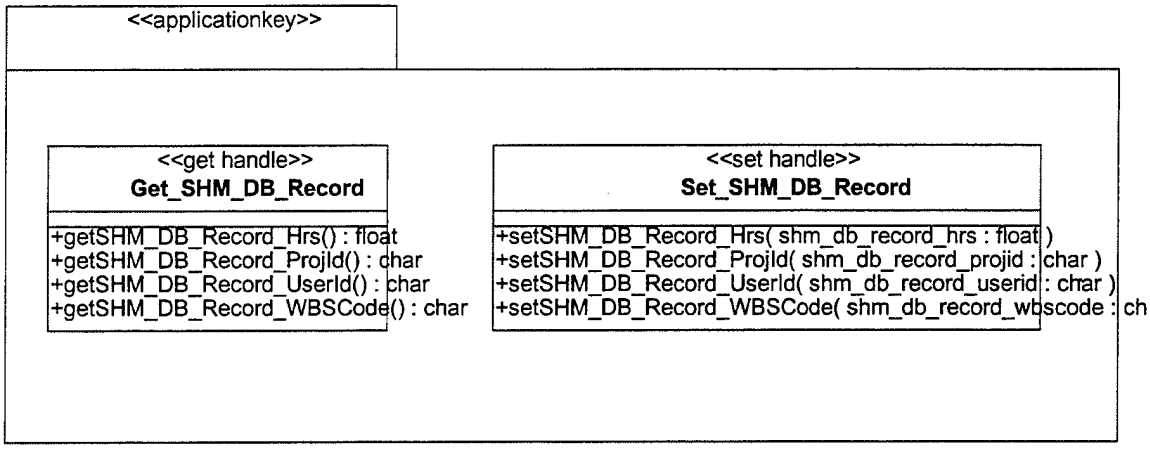


Figure 2-6 Application Keys (Layer 4) after Ref.[6]

5. Enterprise Keys: Enterprise keys form a one-to-one mapping to non-abstract application keys. Users are assigned to enterprise keys at this layer or to a key chain at the next layer. Enterprise keys permit the user to access the methods of the object listed in the key if the application constraints are satisfied. See Figure 2-7.

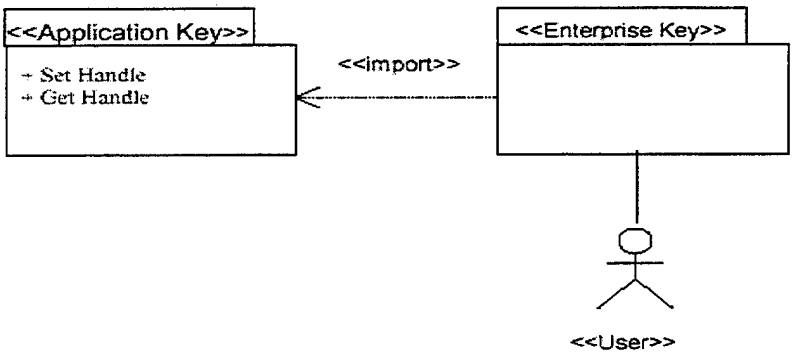


Figure 2-7 Enterprise Key (Layer 5) from Ref.[6]

6. Key Chains: Structures of enterprise keys that allows system administrators to conform to the security policy by assigning users to the appropriate key chains. See Figure 2-8.

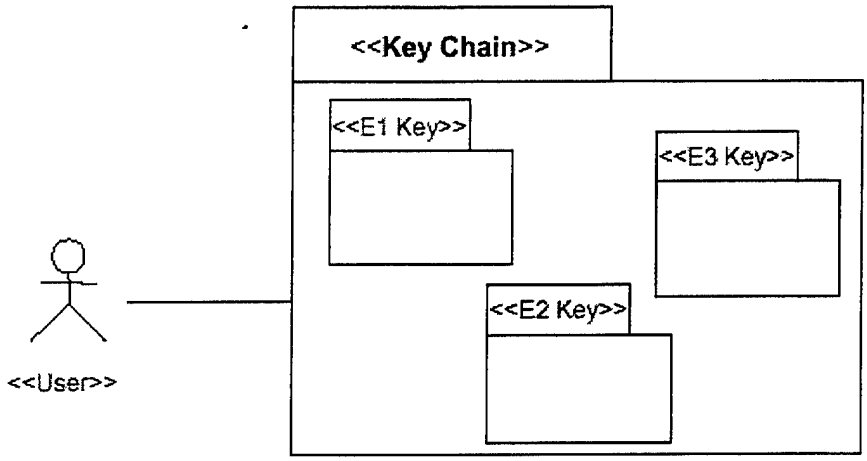


Figure 2-8 Key Chain (Layer 6) from Ref. [6]

7. Enterprise Constraints: These constraints are used to enforce separation of duty policies and restrict unauthorized access to other applications. See Figure 2-9.

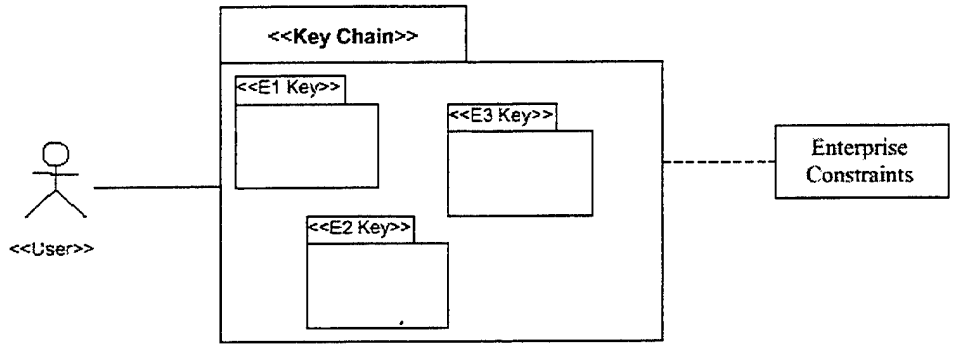


Figure 2-9 Enterprise Constraints (Layer 7) from Ref. [6]

III. SOFTWARE PROCESS MANAGEMENT ROLES AND RESPONSIBILITIES

A. PROCESS DEVELOPER/MAINTAINERS

At SSC San Diego, the Process Developer/Maintainer role is primarily responsible for developing and maintaining the organization's software engineering processes. Processes are developed and changes and improvements are made based on observed results and data collected in the course of process enactment at the project level as shown in Figure 3-1. Process change may also result from changing organization goals and policies. The organization's process developers/maintainers are generally assigned to the Software Engineering Process Group (SEPG). This group uses project and process performance data drawn from the Organizational Software Process Knowledge Base (OSPKB). They analyze process performance data, developing statistical process control charts that provide an indication of whether the processes are statistically stable and performing within desired bounds. Processes that are determined to be statistically unstable or not performing as expected are subject to further analysis and changes are developed to stabilize the processes and bring their performance up to expectations. This is an ongoing effort to continuously improve the organization's software engineering processes. Process developers/maintainers update and manage the configuration of processes and templates in the Process Assets Library (PAL) section of the OSPKB as processes are developed and corrections and improvements are made to existing processes.

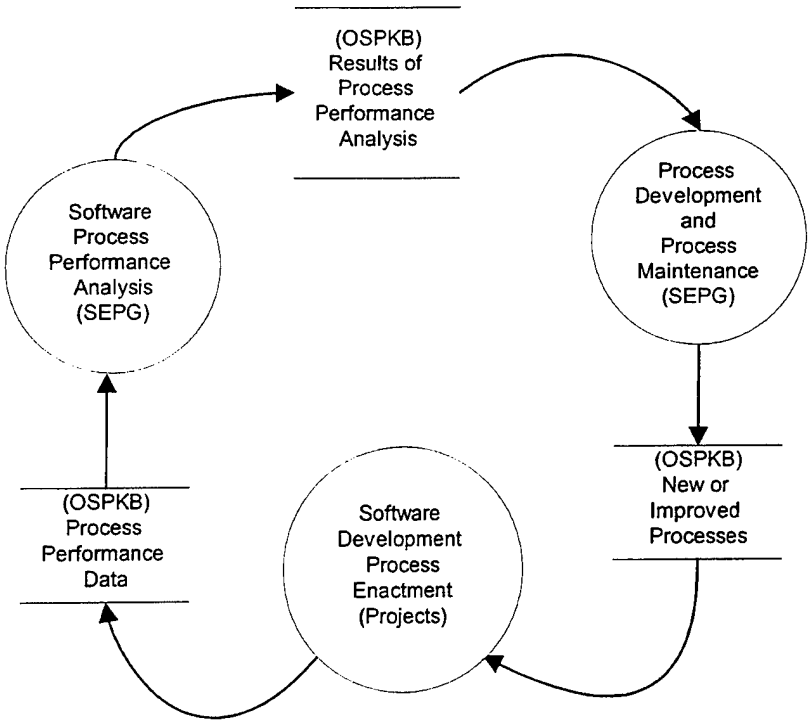


Figure 3-1 Software Process Development/Maintenance

B. PROJECT MANAGEMENT

The Project Management role is responsible for monitoring, controlling, and maintaining project-specific implementations of organizational processes. Process monitoring and controlling responsibilities require the collection and analysis of project performance data at the project level as a result of project execution using the defined processes. A project's defined processes and procedures are changed in response to indicators derived from the statistical analysis of project-specific process performance data, or as a result of changes to the organization's standard processes that the project tailored its specific processes from. Metrics are collected during project execution, and stored in the OSPKB. Metrics analysis software processes the project data in the OSPKB

and generates tracking reports for management. Management alerts are generated when actual results vary by a predetermined threshold from that planned or expected during project execution as shown in Figure 3-2.

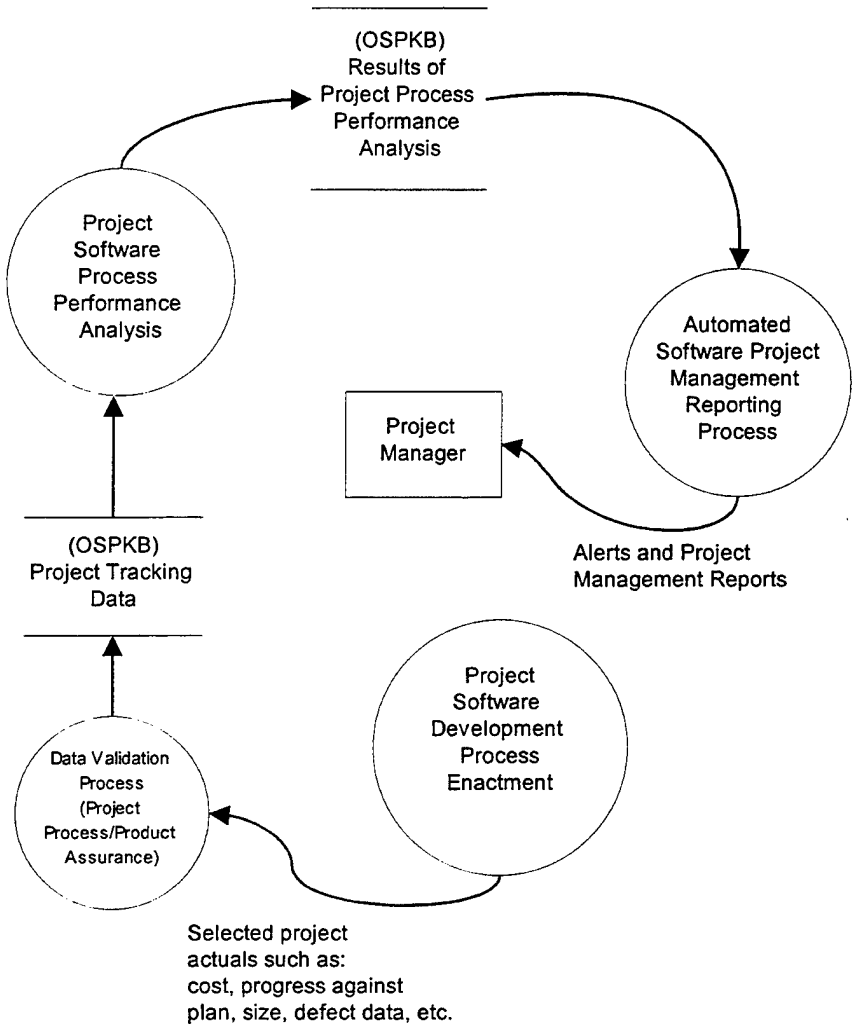


Figure 3-2 Project Management Reports/Alerts

C. PROCESS AND PRODUCT ASSURANCE

The Process and Product Assurance role is responsible for monitoring project execution to ensure both compliance with and proper enactment of the project's defined software

processes. Planning, scheduling and conducting software process and product audits performs this function. Audit results are recorded in the OSPKB as quality audit reports and reported to management.

The Process assurance role is also responsible for performing statistical process monitoring and control functions at the project level, and reporting the resulting process performance data to project management. This group is responsible for validating all project performance data prior to the data being entered into the OSPKB. The product and process assurance roles are depicted in Figure 3-3.

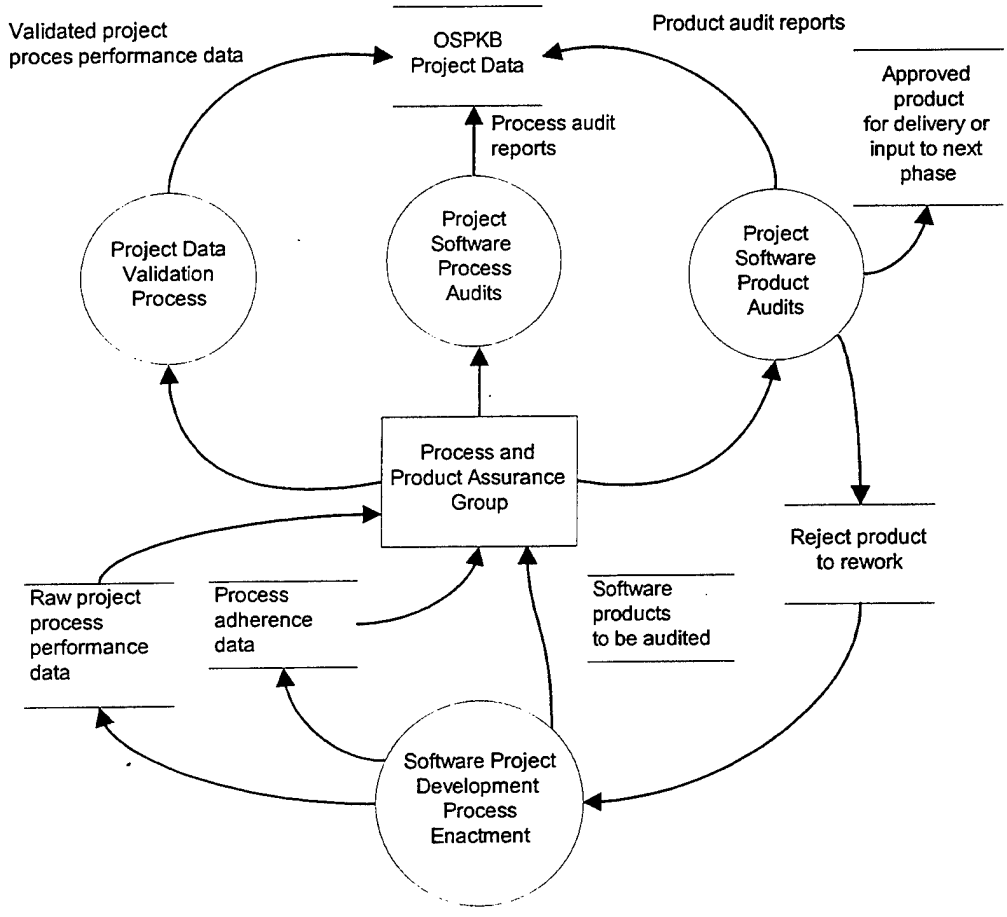


Figure 3-3 Project Process and Product Assurance

D. PROJECT TEAM MEMBERS

Project management and Project Team Members are responsible for enacting the project's defined software engineering processes. Team members use process task checklists, forms, and templates stored in the OSPKB in the carrying out of their individual process roles. While carrying out software development tasks, collection and submission of raw process performance data is an essential responsibility of each project team member. Project team members also provide subjective feedback on process performance to project management. Data is validated and entered into the OSPKB on a regular basis by the Process and Product Assurance group.

Software engineers at the project level generate a large amount of personal process data that is stored in the OSPKB for further analysis by the individual software engineers in the maintenance of their personal software engineering processes. Data from software engineers is in the form of personal plans, estimates, productivity data, personal checklists, defect rates, and other related information. The integrity of this data must be assured and access restricted so the data is not used to improperly compare projects or people. Personal process data in a non-attributable form is used by SEPG personnel at the project and organizational levels to measure planned process performance against actual to assist in the organization's process improvement efforts. Figure 3-4 presents a data-flow diagram of a project team member's interactions with the OSPKB.

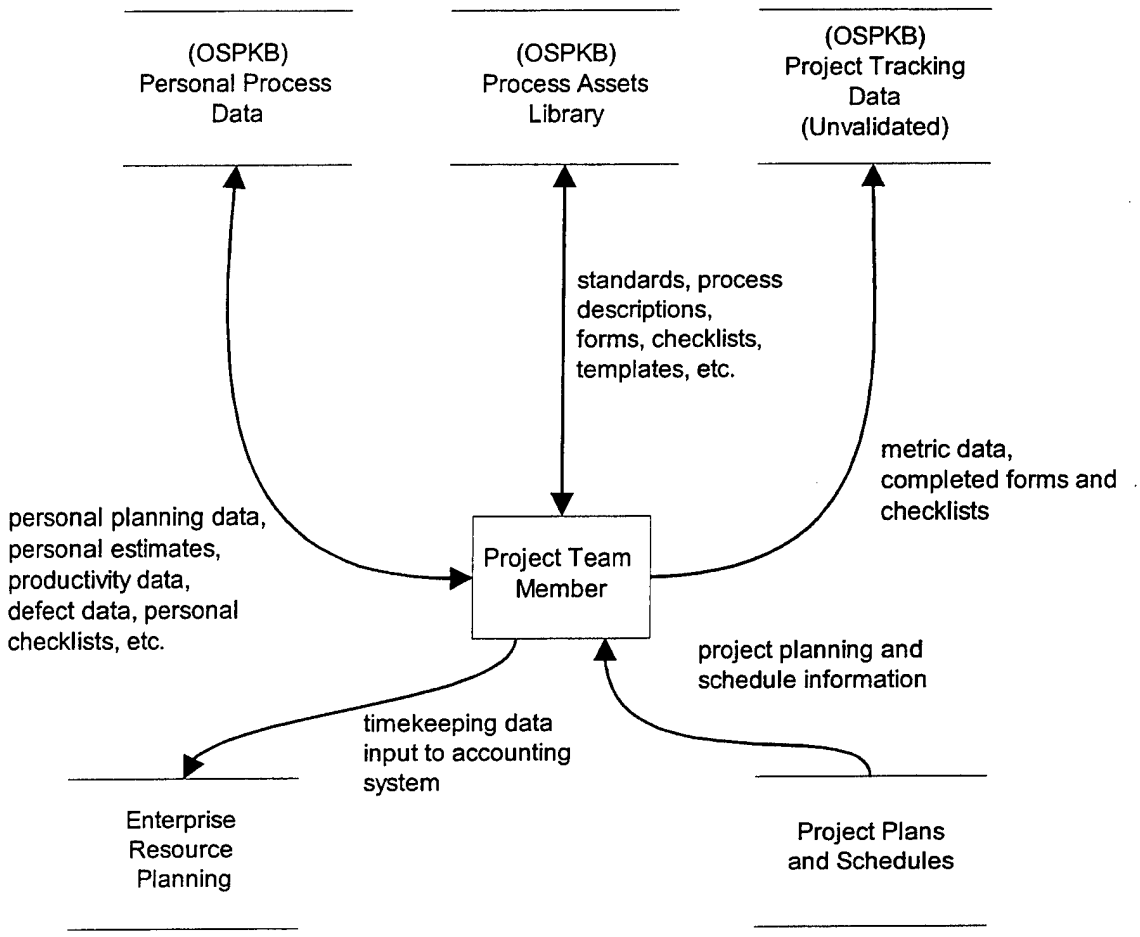


Figure 3-4 Project Team Member Interactions with OSPKB

IV. APPLYING RBAC TO THE OSPKB TO SATISFY SECURITY POLICY

A. OSPKB CONCEPTUAL VIEW

The OSPKB is composed of a federation of related databases and a library of software process assets, as depicted in Figure 4-1.

Access to the OSPKB data is through a set of OSPKB applications that provide the following services:

1. Data Entry/Update functions
2. Query support for data retrieval and analysis
3. Access to the library of documents, templates, forms, and checklists
4. Archival of completed forms and checklists
5. Management report generation

OSPKB applications use RBAC to control access to OSKPB entities and data elements based on user roles, with each role having specific access permissions assigned to it. Each OSPKB user is mapped to one or more roles. Components of the OSPKB are described in the following paragraphs.

The Personal Process component of the OSPKB is a metrics database used by software engineers at the project level. This database contains Personal Software Process (PSP) data collected and maintained on an individual basis. Software engineer roles require create, update, and read access to their individual data. Project-level and Organizational-level process analyst roles require read-only views into the PSP data. Analysts are prevented from attributing PSP data to a particular engineer.

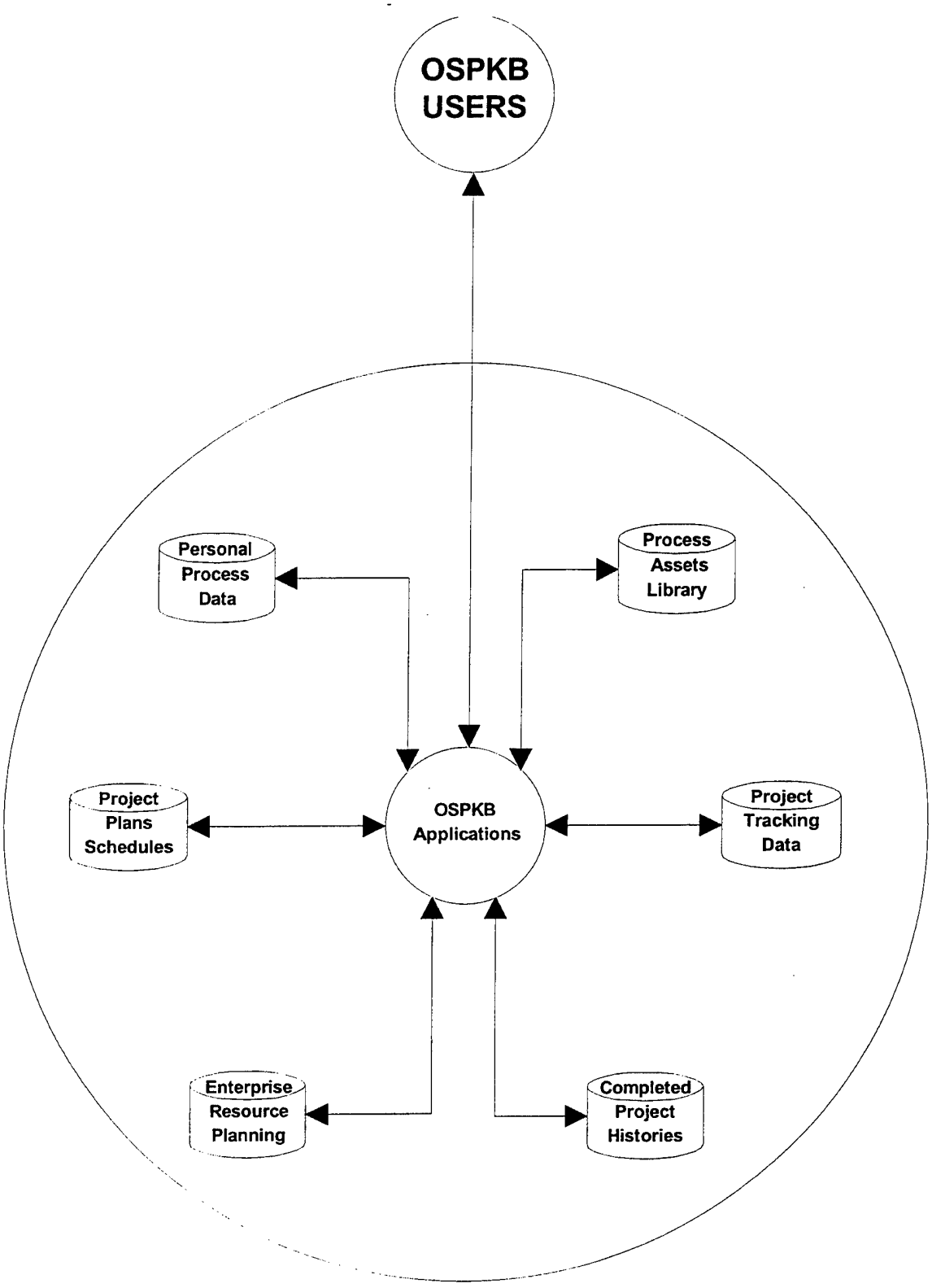


Figure 4-1 Conceptual View of OSPKB

The Project Plans and Schedule component is a database of plans, estimates, and schedules. The data is maintained on a per project basis. Plans and estimates are standard document files. Project schedules are project files in the organization's chosen project scheduling application format. Project analysts require create, update, and read access to their project's data. Organizational analysts require read only access to the data on a non-attributable basis.

The Process Assets Library (PAL) is a web-enabled repository of the process documents, templates, forms, and checklists that implement the organization's Standard Software Engineering Process (SSEP). All user roles require read access to organizational elements of the PAL. The organization's process engineering group requires create and update permissions to the organization portion of the library. Project engineering roles require create, update, and read access to the project portions of the library.

The Project Tracking component is a database of working project data in which actual project size, cost, effort, and schedule data is tracked against baseline plans and schedules. Project personnel require appropriate create, update, and read access to their particular project's data. Project progress reporting roles require read-only access.

Completed project histories is a database of selected data summarizing completed project performance. This database is used to assist project planners in cost and schedule estimation of future projects based on data from similar projects. Data from this component is also used to calibrate automated software estimation tools.

The Enterprise Resource Planning system (ERP) component is the organization's main business system. Each person in the organization is assigned to appropriate ERP

roles that allow them to carry out assigned organizational functions.

B. OSPKB UTILIZATION

The OSPKB is an integral part of the organizational software process management system. Some portions of the system are not automated and are very much manual operations, while other portions are automated to varying degrees. The project planning and estimating process for example, is a manual process. Project planners use the OSPKB as a source of historical data on which to base current project plans and estimates. Once the estimates have been made and project plans and schedules developed the approved plans and schedules are entered into the OSPKB project-plans-and-schedules database. The software project tracking and oversight process (SPTO), an automated process, draws from ERP and project-tracking data that is collected as a product of project execution to produce project progress and earned value reports. The SPTO process generates alerts to project management when progress falls behind schedule or costs exceed budgets by predetermined threshold values. Using RBAC OSPKB applications provide access controls to OSPKB data such that process roles have access to the data required to perform their assigned functions, yet are not permitted to access data that are not associated to those roles.

Process and product assurance processes use completed product review data from the project tracking database to produce product and process quality reports based on actual process performance compared to expected performance. Defect rates, product quality, cost, and schedule performance are all reported as part of the organization's quantitative product and process management processes.

C. OSPKB ROLE HIERARCHY

Role hierarchies are a natural means for structuring roles to reflect an organization's lines of authority and responsibility [5]. A diagram of a role hierarchy for a single project is shown in Figure 4-2. By convention, the most senior roles are shown toward the top of the diagram and the junior roles at the bottom.

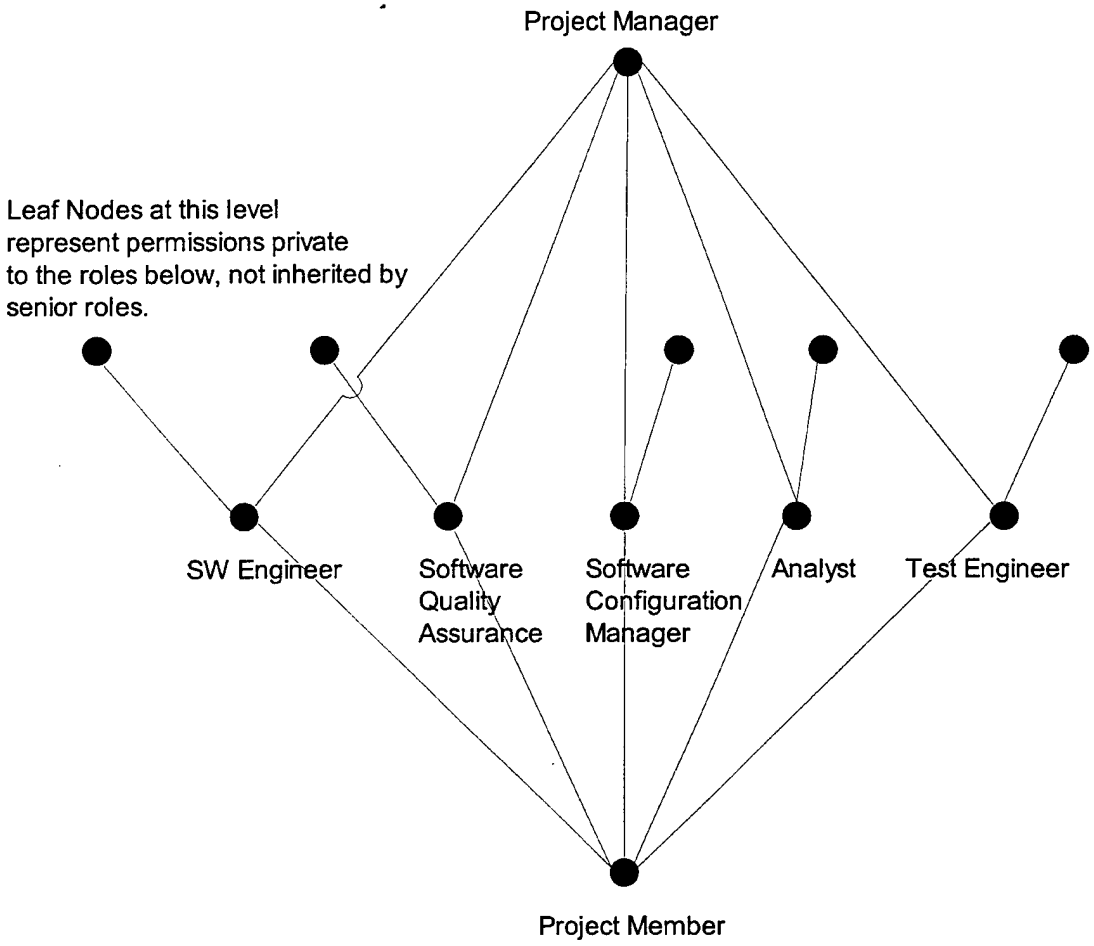


Figure 4-2 OSPKB Project Role Hierarchy

Roles in Figure 4-2 represent groups in a typical project. For example, the SW Engineer node would consist of all software engineers on the project. Each group would have some permissions private to the group and individuals

within the group as shown by the leaf nodes with no connection to a senior role. Similar diagrams with more detail could be drawn for each group, and diagrams with lesser detail could be drawn to depict projects as part of the larger organization.

OSPKB Applications use RBAC satisfy the security requirements for the OSPKB. Data integrity and the level of data confidentiality required by organizational policy are maintained. Reducing the risk of attribution enhances the ability of the organization to collect accurate and timely project measurement data. When personnel don't feel at risk that data will be used/misused against them, they are much more likely to provide accurate data and not provide data that is tainted to be favorable to their position.

D. RBAC APPLIED TO OSPKB APPLICATIONS

Access to OSPKB data objects is provided via basic access methods that are part of the data object. OSPKB applications have access methods with the same names, types and parameters as the methods of the basic access methods object. A role-object layer that lies between the application access methods and the basic access methods object provides access control. The role object class methods have the same names, types and parameters as the application and basic access method classes. The bodies of the role-object class methods contain only conditionals which determine access for the role associated with that role class, and may or may not contain filters which constrain the flow of information between the application interface and the basic access methods. Figure 4-3 shows the layered architecture for implementing RBAC in the OSPKB.

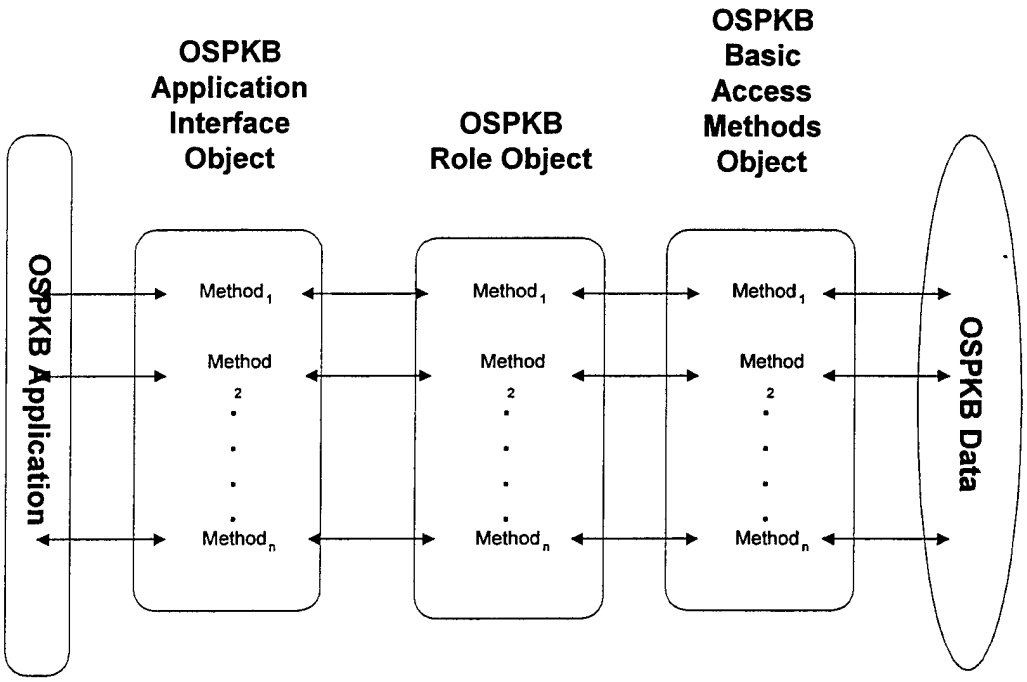


Figure 4-3 OSPKB Data Access Layers after Ref. [9]

The application layer methods invoke the corresponding methods of the role object associated with the current application user, which in turn invokes the methods of the basic access layer if the current role has permission to make the access. The role-object access method will return an error code or message to the application layer if access is not permitted. Each method may be associated with a single data item, or may return an aggregate data structure.

Controlling access in a layered manner allows the access controls to be contained exclusively within the role classes. Applications therefore are not affected by changes in role access policy.

Carrying this concept further, each OSPKB role is assigned a set of role object access methods matching the application interface layer object methods for those applications and basic access methods that the role needs

in order to perform its assigned tasks. The set of basic access objects for each data element equate to the object layer described in the RBAC Framework for Network Enterprises (FNE) Model [7]. The groups of basic access methods specific to a particular data element equate to the object-handle layer described in the model. The access methods assigned to roles equate to the application key layer, and role-specific constraints on data access contained in the role-object access methods equate to the application constraint layer of the model. The enterprise layer of the FNE model is where users are assigned to roles. Application keys are mapped one to one to enterprise keys. Enterprise keys can be assigned to users. Enterprise keys may be grouped into role specific key chains to match organization security policy. Each OSPKB user is assigned a key chain containing the access keys necessary for that user to perform their respective OSPKB role-related responsibilities. Constraints regarding combinations of roles to users are managed at this level. These constraints equate to the enterprise constraint layer of the model.

E. OSPKB APPLICATION AND DATA ACCESS CONTROL

Each person in the organization that is to be a user of the OSPKB is assigned a user role. The user role determines which OSPKB applications and data elements are available to the user during the OSPKB session. Software agents acting on behalf of an OSPKB user have the same role-based access as the user would in performing the same functions. When a user logs into the OSPKB, the user interface application only allows the user to access those applications that are permitted for the user's assigned

OSPKB role. Each application the user accesses uses the layered data access methods described previously to control user access to OSPKB data objects. Specific application and role examples will be described in the following chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

V. OSPKB APPLICATION CASE STUDY

A. STAFF HOUR METRIC APPLICATION

Staff-hour metric collection and utilization is an application that every OSPKB user role is involved with and is part of the project tracking data area of the OSPKB. The Staff-Hour Metric (SHM) database application implements the following RBAC policy:

1. Project members enter only their own staff hour data into the SHM database.
2. Project members may edit only their own data in the database.
3. Project members may generate queries only from their own data.
4. Project Managers and Project Analysts may generate queries to create reports that are not attributable to a particular individual. Data accessed must be from their assigned projects.
5. Division Managers and Division Analysts may generate queries to create reports that are not attributable to a particular individual. Data accessed must be from projects within their division.
6. Senior Managers and Organizational Analysts may generate queries to create reports that are not attributable to a particular individual or project. Data from all projects is available.

The SHM database application uses the role hierarchy shown in Figure 5-1 in implementing the above RBAC policy.

There are four explicit roles in the hierarchy:

1. Senior Manager and Organizational Analyst
2. Division Manager and Analyst
3. Project Manager and Analyst
4. Project Member

The inheritance of junior roles by senior roles also creates implicit roles such as the Senior Manager/Organizational Analyst also being a member of the project member role.

We can formally state the hierarchical relationship as follows:

Let RM = role/membership mapping which gives the set of users authorized for the given role r.

$$(\forall i, j, k, l : role)(\forall u : user)\, i \geq j \geq k \geq l \wedge u \in RM[i] \Rightarrow u \in RM[j] \wedge u \in RM[k] \wedge u \in RM[l]$$

i = Senior Manager/Organizational Analyst

j = Division Manager/Analyst

k = Project Manager/Analyst

l = Project Member

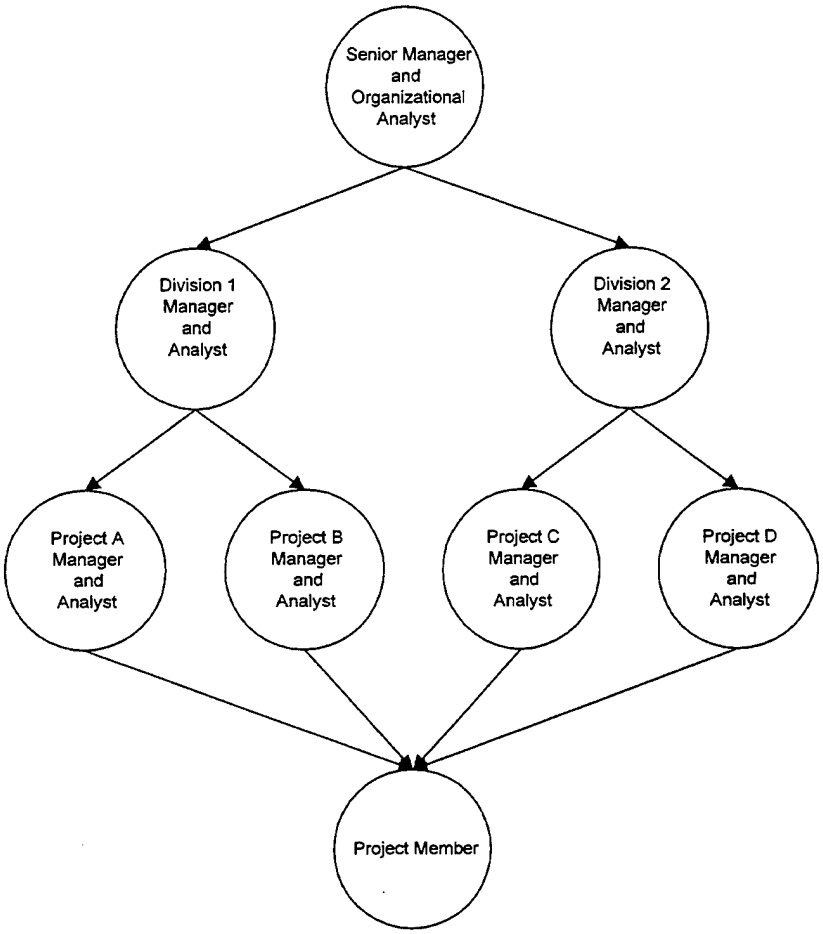


Figure 5-1 SHM Application Role Hierarchy

B. MODELING SHM RBAC

The SHM database application data access interface classes are described by the UML class diagrams in Figure 5-2. The attributes of the interface class are private, and the methods are public. The SHM application calls the methods of the SHM_App_Intfc class when a user accesses the SHM database via the OSPKB SHM user interface. The application interface class uses the UserId parameter to retrieve the proper role access class object. The role access class object has the same exact named public methods. Once the proper role access class object has been retrieved, the application interface class object method calls the corresponding role object's method using the parameters that were originally passed to it. The ProjId

parameter is derived from the user's system login information. The ProjId parameter is passed to limit data access to only the data for the project to which the user is assigned. If all access requirements are satisfied the role access class method calls the basic access class method to perform the actual data access.

The role-class access object's methods provide the conditional processing that limits access to only the data permitted for the user's assigned role, implementing the constraints stated in the RBAC policy for the SHM database.

Implementing access control in the OSPKB using the layered approach described for the SHM application allows changes in security policy to be easily implemented, as only the role access objects need to be modified.

Development of a tool for use by security managers that processes access conditions restricted to filters and conditionals and generates role objects and places them in the dynamically linked libraries (DLL) used by the application would further simplify and enhance the ease with which role/permission mappings are changed.

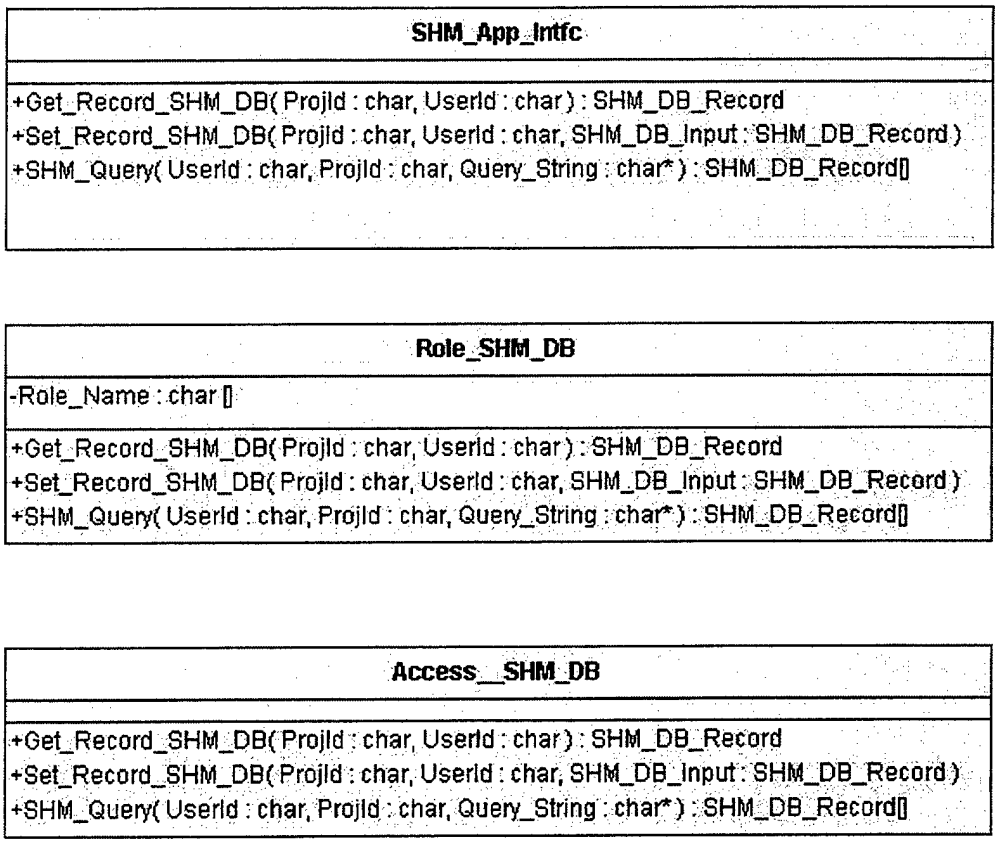


Figure 5-2 Application Interface Classes

C. SHM DATABASE APPLICATION EXAMPLES

In the examples that follow the first six steps in each diagram are the steps that get the user logged into the OSPKB system with the SHM application active. At this point the user's workstation is displaying the SHM GUI.

Figure 5-3 is an example of an individual entering staff-hour metric data into the SHM database. In this example the user selects the SHM data entry function on the user interface. The SHM application then displays the data entry form for the user to fill in. The user fills in the data fields on the form and submits the form. The SHM application then calls the SHM_App_Intfc Set_Record_DB method passing the required arguments. The Set_Record_DB

method calls an OSPKB system procedure (get_role()) that retrieves the users current role. The method then calls the OSPKB system procedure (get_role_obj()) which returns a pointer to the role object for that role. The SHM_App_Intfc method then calls the corresponding Set_Record_DB method in the proper role object passing its input arguments to the role_object method.

The role_object method verifies the userid in the input argument as being the same as the current user to satisfy the stated access condition that project members can only enter their own data. If the condition is not met an exception is raised and the user is notified of the error condition. If the condition is met the role_object method calls the corresponding Set_Record_DB method in the SHM_DB_Access object. The SHM_DB_Access object methods add the new record to the SHM database. Finally a copy of the newly entered record is returned via the access objects to be displayed to the user.

The data update function of the SHM database application shown in Figures 5-4 and 5-5 implements the condition that states project members can update only their own data. This is accomplished in a manner identical to data entry. The role object method checks to see if the data requested by the update query refers to the same userid as the user making the request. If the condition is met the update is made as in Figure 5-4, if not, an exception is raised and an error message is displayed to the user and the update is not made as in Figure 5-5.

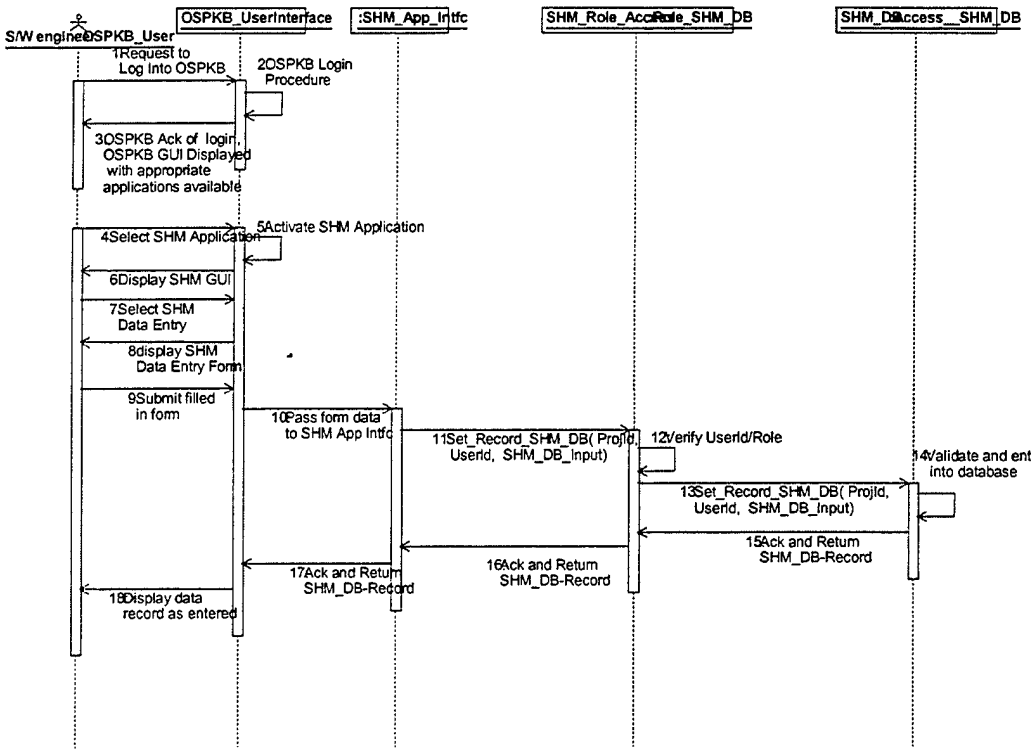


Figure 5-3 SHM_DB Data Entry Sequence Diagram

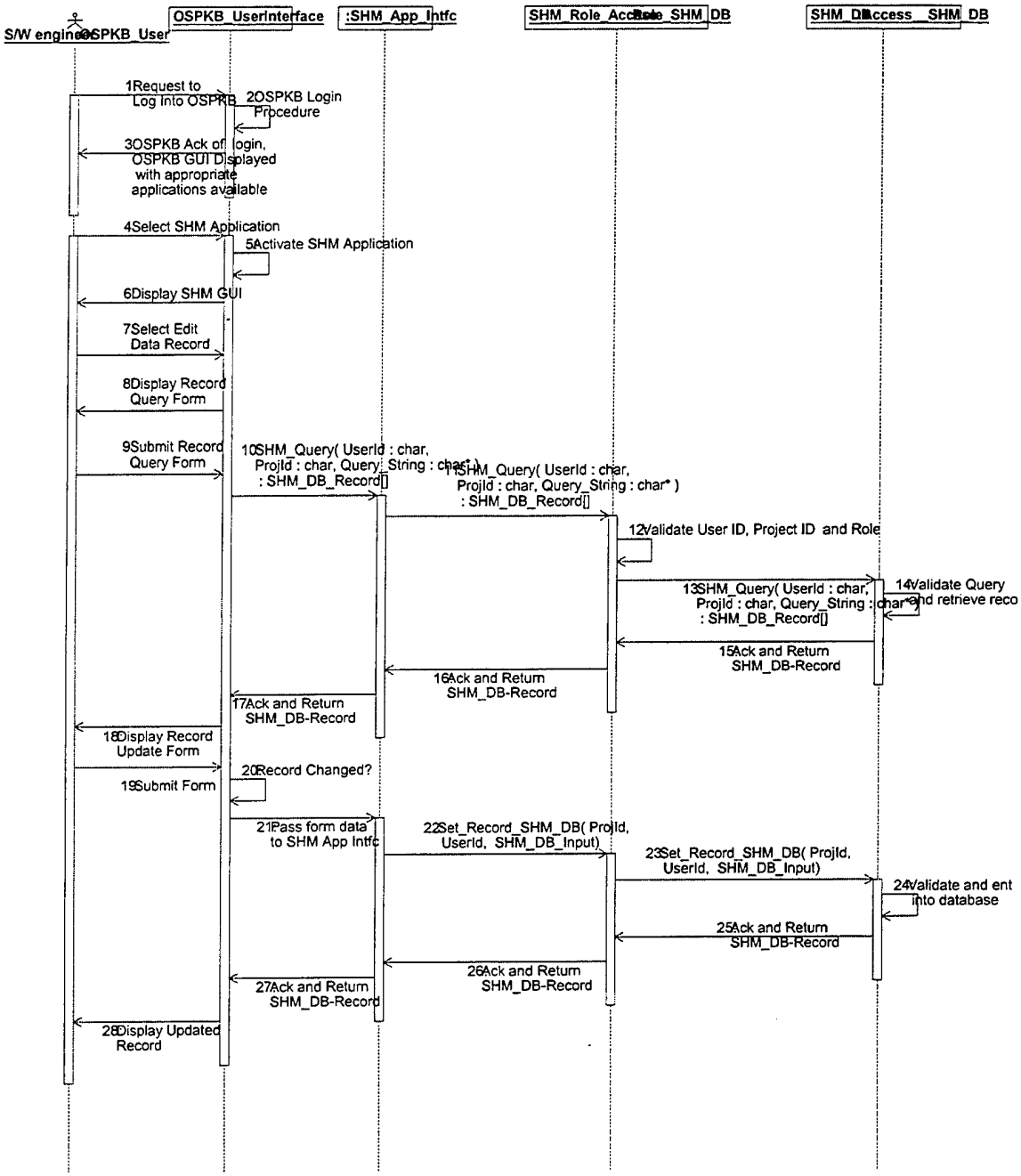


Figure 5-4 SHM_DB Data Update Sequence Diagram

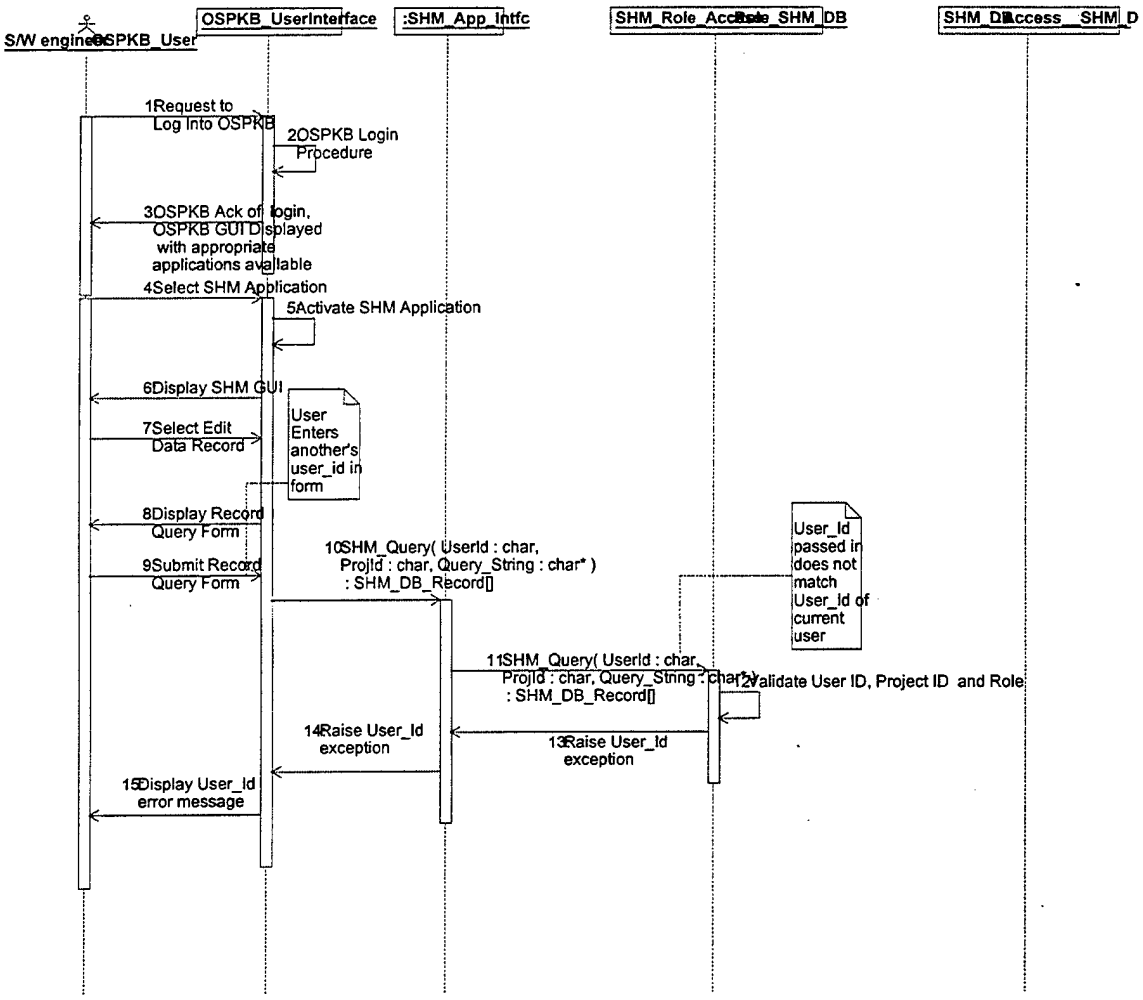


Figure 5-5 Data Update Error Sequence Diagram

Figure 5-6 shows the interaction when a project manager/analyst requests a report for SHM data for his project. The access conditions of project managers/analysts only being allowed to query their own project data is satisfied when the role access object method verifies the project id in the database query matches the id of the manager/analyst's currently assigned project. The constraint that requires the data returned to be non-attributable to a particular user is satisfied by the role access object method filtering out the user data from the record set returned by the database query. Figure

5-7 shows the interaction when the project id requested in the query does not match the project manager/analyst's currently assigned project id. The role access object method raises an exception resulting in an error message being displayed to the user and the query not being allowed.

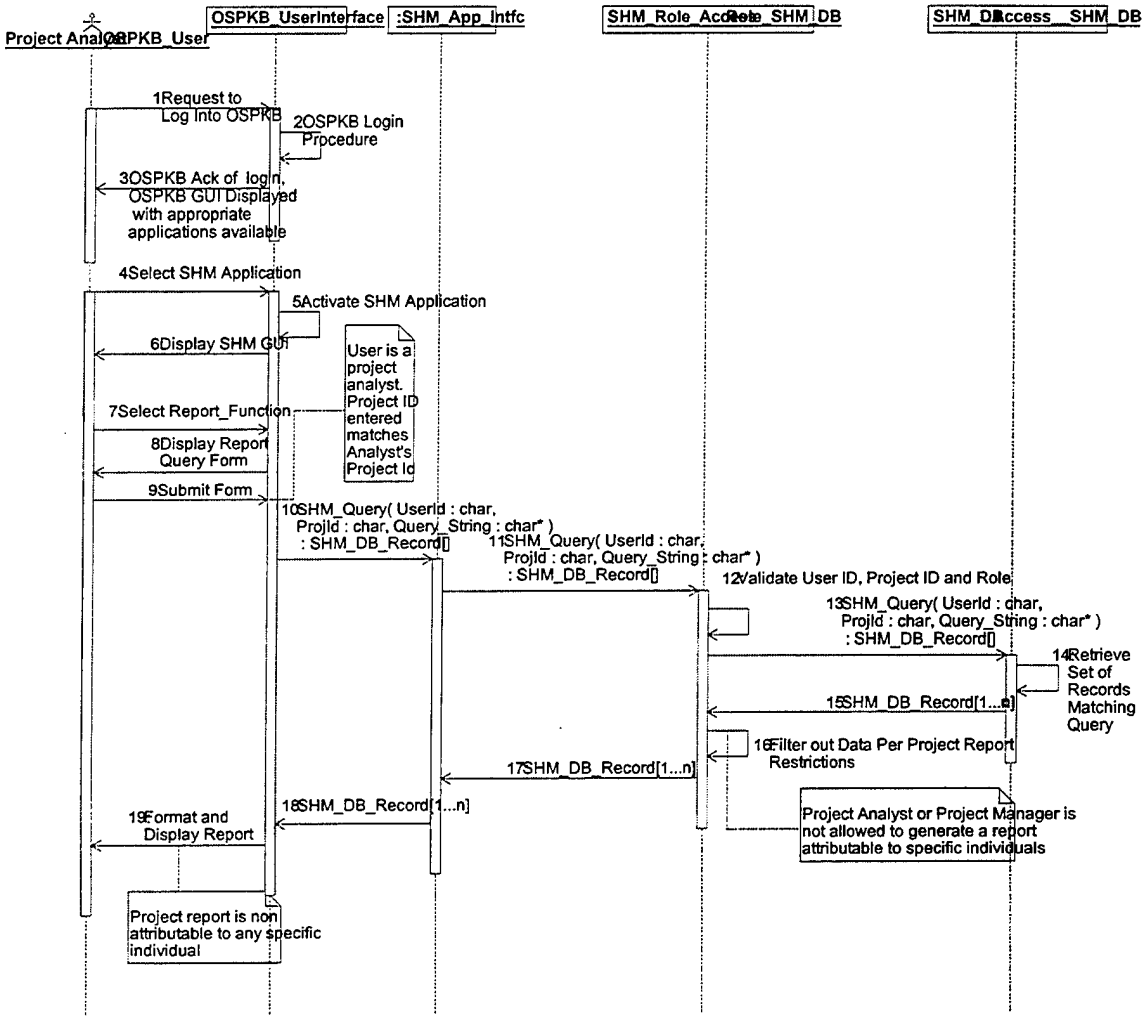


Figure 5-6 Project Manager/Analyst Report Sequence Diagram

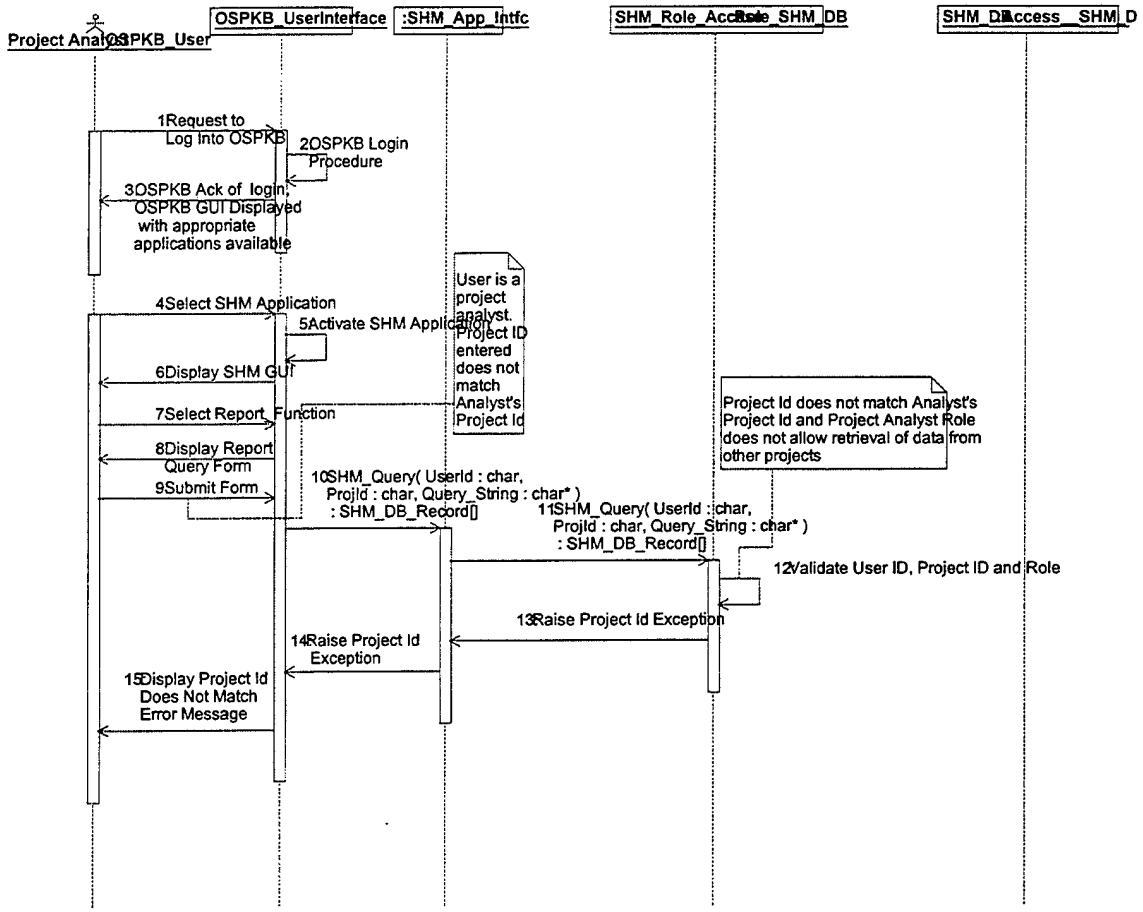


Figure 5-7 Project Manager/Analyst Report Error Sequence Diagram

Division level and Organization level roles are handled in an identical manner. Access conditions are satisfied by the conditionals in the role access object methods. Constraints are satisfied by the data filters incorporated in the role access methods.

D. DISCUSSION

Implementing security with RBAC in the manner described greatly reduces the cost and complexity of security administration for the OSPKB.

This cost reduction is articulated by contrasting the administrative cost complexity of direct user to permission

associations with that of an RBAC solution. An OSPKB using direct user/permission association for a large organization with many users, each with many permissions, requires a large number of user/permission associations. When a user switches positions within the organization, a thorough review, addition, and deletion of user/permission associations on each server is required. There is a risk of residual and inappropriate user access rights.

Administrative cost complexity can be quantified as follows:

U = Set of individuals in a job position

P = Set of OSPKB permissions required for that job position

|U|*|P| = The number of associations required to directly relate the individuals to permissions

In contrast, RBAC does not permit users to be directly associated with permissions, rather permissions are authorized for roles, and roles are authorized for user.

RBAC requires the administration of two associations:

1. Between users and roles
2. Between roles and permissions

With RBAC when a user switches positions within the organization only two changes are required:

1. Remove OSPKB user/role association for old position
2. Insert OSPKB user/role association for new position

Risk of residual or inappropriate user access rights is greatly reduced.

RBAC administrative cost complexity can be quantified as follows:

U = Set of individuals in a job position

P = Set of OSPKB permissions required for that job position

|U|+|P| = The number of user/role and role permission associations required to authorize each user in set U for

each of the permissions in the set P (P represents the OSPKB role). For a given number of OSPKB roles, |U|+|P| is much less than |U|*|P|.

The SHM database case study described in this thesis is a subset of the total OSPKB. The example is representative of role-based access control in all of the OSPKB applications. Different role hierarchies would be associated with the various OSPKB applications.

Users associated with roles in the OSPKB may be a human or possibly an intelligent software agent acting on behalf of a human.

Lori Church describes a software process management system conceptual model called MENTOR in her Master's thesis [10]. MENTOR is based on using intelligent software agents as members of the software development team. Mentor's agents take on many different organization roles as they perform their software engineering functions. Mentor makes extensive use of databases as repositories of software engineering information. The OSPKB described in this thesis using RBAC for security could fulfill many of the MENTOR database requirements at the organization. MENTOR agents associated to OSPKB roles would interact with OSPKB applications to perform their MENTOR functions. Agent access to data would be controlled by the RBAC policy and role hierarchies established for each OSPKB application.

Security and privacy of information, in an agent-based system such as MENTOR, is of great concern. RBAC provides a practical solution to the problem. MENTOR would benefit from the economies of security administration costs realized in an RBAC based security system. Use of access object technology [9] layered access approach is compatible with the agent architecture used in the MENTOR system. MENTOR could use centralized databases to store the RBAC

role sets, agent to role associations, and the library of role access objects that implement the controls, allowing centralized control and administration of RBAC implementation. When an application needs a particular role access class object it could retrieve it from the central database.

VI. CONCLUSIONS

By administering the granting and revocation of access permissions via RBAC, security management is by role rather than by individual user. This simplifies organizational security management by significantly reducing the complexity of administering the security of the OSPKB by assignment of access permissions to a predefined set of OSPKB roles. The predefined role set could be stored in a role database. When a user is assigned to a position or departs the organization, the security manager reassigns the user to the roles defined for the new position or removes the user from the role assignment system. User requests for access to OSPKB data via OSPKB applications are arbitrated by appropriate role access method objects that satisfy the disclosure policies established for OSPKB data.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. FUTURE WORK

In this thesis we have shown by example how RBAC can be applied to OSPKB applications.

A. REQUIREMENTS ANALYSIS

A logical step to further the development of an OSPKB that employs RBAC would be to develop a set of detailed requirements for an OSPKB that includes security and privacy. This thesis only deals with the conceptual model of an OSPKB; therefore detailed requirements remain to be developed.

B. PROTOTYPE DEVELOPMENT

Development of a working prototype OSPKB would be another logical step. The prototype should include the main OSPKB user interface and application user interfaces. The prototype should assist in the integration of the OSPKB components.

C. APPLICATION DEVELOPMENT TOOLS

To aid in OSPKB application development, a set of tools could be developed that would automate the generation of the role access method classes based on a set of class templates for each application implementing the application's RBAC policies.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Humphrey, Watts, *Managing the Software Process*, Addison Wesley, 1990.

2. Deming, W.E., *Quality, Productivity, and Competitive Position*, Cambridge, MA: Massachusetts Institute of Technology, 1982.

3. Florac, William A., Park, Robert E. and Carleton, Anita D., Practical software Measurement: Measuring for process Management and Improvement, Software Engineering Institute, Carnegie Mellon University CMU/SEI-97-HB-003, 1997.

4. Montgomery, Douglas C., *Introduction to Statistical Quality Control*, 3rd ed., New York, N.Y.: John Wiley & Sons, 1996.

5. Sandhu, Ravi S., "Role-Based Access Control," in *Advances in Computers*, vol. 46, Academic Press, 1998.

6. Eptstein, Pete and Sandhu, Ravi S., "Towards a UML-Based Approach to Role Engineering," *Proceedings of the 4th ACM Workshop on Role-Based Access Control*, pp. 135-143, 1999.

7. Thomsen, Dan, O'Brien, Dick, and Bogle, Jessica, "Role Based Access Control Framework for Network Enterprises (FNE)", *Proceedings of 14th Annual Computer Security Conference*, pp. 50-58, 1998.

8. Booch, Grady, Rumbaugh, James, and Jacobson,Ivar, *The Unified Modeling Language User Guide*, Addison Wesley Longman, Massachusetts, 1999.

9. Barkley, John, "Implementing Role Based Access Control using Object Technology," *Proceedings of the First ACM Workshop on Role-Based Access Control*, pp. 1193-1198, November 1995.

10. Church, Lori A., *Decision Support for Software Process Management Teams: An Intelligent Software Agent Approach*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 2000.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center.....2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library.....2
   Naval Postgraduate School
   411 Dyer Road
   Monterey, California 93943-5100

3. Chairman, Code CS.....1
   Naval Postgraduate School
   Monterey, California 93943-5100

4. Dr. Luqi, Code CS/Lq.....1
   Computer Science Department
   Naval Postgraduate School
   Monterey, California 93943-5100

5. Dr. J. Bret Michael, Code CS/Mj.....1
   Computer Science Department
   Naval Postgraduate School
   Monterey, California 93943-5100

6. Dr. John Osmundson, Code AG/Oj.....1
   C3 Academic Group
   Naval Postgraduate School
   Monterey, California 93943-5100

7. Dr. Man-Tak Shing, Code CS/Sh.....1
   Computer Science Department
   Naval Postgraduate School
   Monterey, California 93943-5100

8. Systems Engineering Process Office, Code D12.....1
   SPAWARSYSCEN San Diego
   53560 Hull Street
   San Diego, California 92152-5100

9. Dr. Ravi Sandhu.....1
   Department of Information and Software Engineering
   George Mason University
   Mail Stop 4A4
   Fairfax, VA 22030-4444

10. Mr. William Windhurst..... 1
1763 Tamarand Way
San Diego, California 92154-2858